CRT-2000-29

Centre de recherche sur les transports (C.R.T.)
Centre for Research on Transportation

A SIMPLEX-BASED TABU SEARCH METHOD FOR CAPACITATED NETWORK DESIGN

by

Teodor Gabriel Crainic
Michel Gendreau
Judith M. Farvolden

September 2000

101-01797/

Revised version of CRT-98-37.
Published in INFORMS Journal on Computing 12(3), 223–236, 2000.

PUBLICATION
CRT-2000-29

*A SIMPLEX-BASED TABU SEARCH METHOD FOR CAPACITATED NETWORK DESIGN**

by

Teodor Gabriel Crainic[1,2]
Michel Gendreau[2,3]
Judith M. Farvolden[4]

* *This research was supported in part by the Fonds pour la formation des chercheurs et l'aide à la recherche (F.C.A.R.) and by the Natural Sciences and Engineering Research Council of Canada (NSERC).*

[1] Département management et technologie, Université du Québec à Montréal, C.P. 8888, succursale A, Montréal, Canada H3C 3P8

[2] Centre de recherche sur les transports, Université de Montréal, C.P. 6128, succursale Centre-ville, Montréal, Canada H3C 3J7

[3] Département d'informatique et de recherche opérationnelle, Université de Montréal, C.P. 6128, succursale Centre-ville, Montréal, Canada H3C 3J7

[4] Algorithmics Inc., 822 Richmond Street W., 3rd floor, Toronto, Canada M6J 1C9

Département d'informatique et de recherche opérationnelle — Publication #1180

Dépôt légal – Bibliothèque nationale du Québec,
Bibliothèque nationale du Canada, 2000

Abstract

The fixed charge capacitated multicommodity network design problem is a well-known problem, of both practical and theoretical significance. This paper presents an efficient procedure to determine tight upper bounds on the optimal solution of realistically sized problem instances. Feasible solutions are obtained by using a tabu search framework which explores the space of the continuous flow variables by combining pivot moves to column generation, while evaluating the actual mixed integer objective. Computational experiments on a large set of randomly generated test problems show that this procedure outperforms the other available methods and is particularly suited to large problem instances with many commodities.

Keywords: fixed charge capacitated multicommodity network design, tabu search, mixed integer programming, column generation

Résumé

Le problème de synthèse (design) de réseau multiproduit avec coûts fixes et capacités présente un grand intérêt scientifique et pratique. Dans cet article, nous présentons une méthode efficace pour déterminer des bornes supérieures serrées sur la valeur optimale de tels problèmes, même lorsque ceux-ci sont de grande taille. Des solutions réalisables sont obtenues grâce à une méthode de recherche adaptative avec tabous combinant des mouvements de pivotage, évalués à partir de l'objectif original entier mixte, avec de la génération progressive de colonnes. Les résultats expérimentaux obtenus sur un grand nombre de problèmes générés aléatoirement, confirme que cette procédure est la plus performante actuellement et traite facilement des problèmes de grande taille avec un nombre important de produits.

Mots clé: synthèse (design) de réseau multiproduit avec coûts fixes et capacités, méthodes de recherche adaptative avec tabous, programmation entière mixte, génération progressive de colonnes

1 Introduction

The fixed charge network design problem is a well-known problem, of both practical and theoretical significance [1]. We address, in particular, the fixed charge, capacitated, multicommodity network design problem with linear costs (CMND). This class of formulations appears prominently when addressing issues in infrastructure construction or improvement, service network design, telecommunication network design and dimensioning, power system design, etc. Here, several "commodities" (goods, data packets, people,...) have to be moved over the links of a network with limited capacities from their respective origins to their respective destinations. Furthermore, other than the usual "transportation" costs related to the volume of each commodity flowing through a given link, a "fixed" (construction or utilization) cost is payed as soon as a link is used. The trade-off between the variable and fixed costs inherent in the selection of any given solution, as well as the interplay between the limited capacity (and the resulting competition among the various commodities) and the fixed costs of using the links of the network, makes for a formulation which presents both a broad modelling appeal and serious obstacles when one attempts to solve realistically sized instances.

Interesting results have been derived for the uncapacitated design problem [28, 29, 5, 27]. Less effort has been directed towards the capacitated problem, which is more difficult to solve and poses considerable algorithmic challenges [4]. See [26, 32, 31, 3, 22], for previous attempts at addressing these difficult problems. Recently, Gendron and Crainic [15, 16] (see also [14, 17]) have derived tight lower bounds for the general class of capacitated, multicommodity, fixed charge network flow problems, but their upper bounds, for now, are disappointing. As for sequential branch-and-bound, it can not solve realistically dimensioned cases. For capacitated problems of any interesting size, only specially tailored heuristics have proved of any help. The simplest of these are drop and add procedures based on reduced cost calculations which determine the marginal value of including or excluding an arc from the network [30, 25], while the more sophisticated make use of the marginal values of paths in the network [11, 23, 12]. These local search heuristics were applied to particular problem classes, however, and did not attempt to explore past the first local optimum. Moreover, they generally required significant computation times.

The goal of this paper is to present an efficient procedure to find good feasible solutions to realistically sized capacitated multicommodity fixed cost network design problems. The procedure is a Tabu Search metaheuristic [21] which combines simplex pivot type moves with column generation to produce a search that explores the space of the continuous path flow variables, while evaluating the actual mixed integer objective of the CMND. The experimental results that we report show that the procedure is robust with respect to the choice of parameter values, and that it finds very good solutions to large problems with many commodities. In fact, to our knowledge at least, this is actually the best procedure for finding good feasible solutions for the CMND.

The contribution of this paper is twofold. First, it provides an efficient heuristic for a difficult network optimization problem. Second, it expands our understanding of the tabu search metaheuristic by studying the relationships between the tabu search framework, simplex pivoting, and column generation.

The paper is organized as follows. Section 2 presents formulations of the problem and the basic ideas of the method. The tabu search method is detailed in Section 3. Calibration and experimental results are reported and analyzed in Sections 4 and 5, respectively. We conclude with a few general remarks and possible avenues to continue this work.

2 Formulations and algorithmic idea

Let $\mathcal{G} = (\mathcal{N}, \mathcal{A})$ be the graph underlying the network to be designed, $\mathcal{N}$ the set of nodes, $\mathcal{A} = \{a = (i,j) \mid i \in \mathcal{N}, j \in \mathcal{N}\}$ the set of arcs (for the sake of simplicity, we assume all arcs to be directed design arcs), and $\mathcal{P}$ the set of commodities to be distributed. Without loss of generality, we define a commodity $p \in \mathcal{P}$ to be the flow between an origin node $r \in \mathcal{N}$ and a destination node $s \in \mathcal{N}$, while the corresponding demand is denoted w^p.

Three measures characterize each arc $a \in \mathcal{A}$: c_a^p, the unit cost of moving commodity $p \in \mathcal{P}$ through the link, f_a, the fixed cost of including the arc in the design of the network (or to introduce additional capacity on the arc), and u_a, the total capacity of the arc. The flow of commodity $p \in \mathcal{P}$ moves from its origin r^p to its destination s^p by using one or several paths in set $L^p = \{\text{all paths from } r^p \text{ to } s^p \text{ in } \mathcal{G}\}$. According to standard network notation, $\delta_{al}^p = 1$ if arc a is in the l^{th} path for commodity p and 0 otherwise, while the cost of path $l \in \mathcal{L}^p$ is $k_l^p = \sum_{a \in \mathcal{A}} c_a^p \delta_{al}^p$.

The decisions concern the selection of arcs for the final design of the network — $y_a = 1$ if arc $a \in \mathcal{A}$ is included (opened) and 0 otherwise —, and the path volumes to satisfy the demand: h_l^p, the flow of commodity $p \in \mathcal{P}$ on path $l \in \mathcal{L}^p$. The corresponding arc flows, x_a^p, may then be obtained by the usual computation $x_a^p = \sum_{l \in L^p} h_l^p \delta_{al}^p$.

The general form of the path–based formulation of the fixed-cost, capacitated multi-commodity network design problem (PCMND) may then be written as:

$$\text{Minimize} \quad z(h,y) = \sum_{a\in\mathcal{A}} f_a y_a + \sum_{p\in\mathcal{P}}\sum_{l\in\mathcal{L}^p} k_l^p h_l^p \tag{1}$$

$$\text{Subject to} \quad \sum_{l\in\mathcal{L}^p} h_l^p = w^p \qquad \forall p \in \mathcal{P}, \tag{2}$$

$$\sum_{p\in\mathcal{P}}\sum_{l\in\mathcal{L}^p} h_l^p \delta_{al}^p \leq u_a y_a \qquad \forall a \in \mathcal{A}, \tag{3}$$

$$h_l^p \geq 0 \qquad \forall p \in \mathcal{P}, l \in \mathcal{L}^p, \tag{4}$$

$$y_a = \{0,1\} \qquad \forall a \in \mathcal{A}. \tag{5}$$

It is well known that, for any set of values $\tilde{y}$ of the design variables y, this path–based (equations 1 to 5) formulation of the CMND becomes a capacitated multicommodity minimum cost network flow (MCNF) problem in path flow variables [1] (the same relation also holds for arc-based formulations). The objective of such formulations is to distribute a number of items between their points of origin and destination, over a network in which some or all of the arcs have limited capacity. Capacitated multicommodity network flow problems are linear programs and a rich body of work addresses the issue of how to solve them efficiently [2, 24, 1]. The solution approach that we propose for the multicommodity capacitated network design problem is based on a set of very interesting relations which follow from this fundamental connection between the design and minimum cost formulations.

Let $\mathcal{H}(\tilde{y})$ denote the set of all feasible path flows in PCMND for a given $\tilde{y}$. Then, $\mathcal{H}(\tilde{y})$ is also the set of all feasible path flows for the corresponding MCNF and an optimal solution to this problem can always be found at an extreme point of $\mathcal{H}(\tilde{y})$, assuming $\mathcal{H}(\tilde{y}) \neq \emptyset$. Denote by $\bar{\mathcal{H}}(\tilde{y})$ the set of such extreme points. Consider now $\mathcal{H} = \mathcal{H}(\tilde{y} = 1)$, the set of feasible path flows when all the design arcs are open (i.e., $\tilde{y}_a = 1,\ \forall a \in A$), and its associated set of extreme points $\bar{\mathcal{H}}$. $\mathcal{H}$ is defined by the set of constraints 2, 3, and 4 and it is easily verified that $\bar{\mathcal{H}} = \bigcup_{\tilde{y}\in Y} \bar{\mathcal{H}}(\tilde{y})$, where Y is the set of all possible design variable vectors. Consider, finally, that for any given path flow pattern $\tilde{h} \in \bar{\mathcal{H}}$, a design with minimal cost relative to the path flows, denoted $y(\tilde{h})$, can be obtained by setting

$$y_a(\tilde{h}) = \begin{cases} 0 & \text{if } \sum_{p\in\mathcal{P}} \sum_{l\in\mathcal{L}_a^p} \delta_{al}^p \tilde{h}_a^p = 0 \\ 1 & \text{otherwise} \end{cases} \tag{6}$$

where L_a^p stands for the set of paths for product $p \in \mathcal{P}$ that use the design arc $a \in \mathcal{A}$, $\mathcal{L}_a^p \subset \mathcal{L}^p$. An optimal solution to the CMND can therefore be found by complete or implicit enumeration of the elements of $\bar{\mathcal{H}}$.

A solution procedure for the CMND based on enumerating the elements of $\bar{\mathcal{H}}$ is likely to be extremely time-consuming, however, even for moderately-sized instances. Alternatively, the previous property may be used to define local search heuristics for the

CMND: the adjacency relationships in $\tilde{\mathcal{H}}$ define a natural neighbourhood structure, while the pivoting rules of the simplex method provide an efficient way to reach the neighbours of any given solution. Thus, the fundamental idea of the approach we propose for solving the CMND is to use a tabu search method [21] to explore the solution space $\tilde{\mathcal{H}}$, where the revised simplex algorithm offers the framework within which the neighbourhood is constructed and explored, and moves are evaluated, selected and implemented.

Problems of practical interest are usually too large, however, to be solved in reasonable time by standard linear programming techniques, and specialized solution methods have to be developed. In particular, Farvolden, Powell and Lustig [13] exploited the structure of the constraint matrix (equations 2 and 3) to derive an efficient specialization of the primal simplex method for the path–based formulation of the MCNF. The algorithm is based upon primal partitioning and Dantzig-Wolfe decomposition. To partition the basis, basic slack variables (in the capacity constraints 3) are separated from basic flow variables, which are further divided into path variables for unsplit (one path carries all the flow of the given commodity) and split commodities, respectively. As for rows, these are partitioned into flow conservation constraints for unsplit commodities, the same constraints for split commodities plus the capacity constraints for saturated links, and, finally, the capacity constraints for the unsaturated links. This yields a working basis which is, in general, very small compared to the complete basis. Moreover, several pivot types and dual variable computations may be defined analytically. Thus, the majority of simplex operations performed on the partitioned basis are simply additive and corresponded to network operations adapted to these particular pivot types. Only operations on the sparse, near-triangular working basis require matrix algebra. Finally, to overcome the difficulty of the prohibitive number of paths associated with real-life problems, the authors resorted to a column generation scheme: the method starts with a small number of paths for each commodity and new paths are added as the algorithm proceeds. The new paths are obtained by solving a shortest path problem on the associated graph, where arc costs are the reduced costs relative to the current basis. Refer to [13] for a complete description and analysis of the algorithm.

We make use of the same linear programming ideas to develop the tabu search procedure. The algorithm exhibits the main structure of the primal simplex method with basis partitioning and column generation. Other than its intrinsic qualities in terms of efficient identification and treatment of particular pivot types, this approach offers two significant advantages:

- At all times, either the current solution of the flow problem is feasible or a feasible solution, if one exists, may be reached easily by using dual steps;
- Good quality variables (measured in terms of marginal costs), and therefore candidate moves, may be easily generated in a controlled/limited number through column generation techniques.

Adapting the primal simplex with column generation to fit within a tabu search framework represents an interesting challenge, however, since several of the standard assumptions upon which column generation schemes are based are no longer verified. In particular, the monotonic decrease of the objective function value is no longer ensured. Therefore, an exact stopping criterion no longer exists. Moreover, the guarantee that new path variables are different from those already generated no longer applies. Also, since variable and fixed costs characterize arcs, both have to be included in the pivoting and column generation schemes. Th following section details how we address these issues.

3 Tabu Search for Network Design

Tabu search is a metaheuristic [18, 19, 20, 21] designed to escape the trap of local optimality. At its core, usually there is a local search scheme that explores iteratively the space of potential solutions by moving from one of these solutions to another at each step. To each solution is associated a *neighbourhood* consisting of all solutions which may be reached from the current solution by means of a transition called a *move*. At each iteration, the neighbourhood of the current solution is examined and the candidate yielding the best reward (the objective function value or some other quality measure) is selected to become the new current solution. To avoid cycling, a recent history of the search is maintained in constantly updated short term *tabu memories*; this information is used to prevent the search from returning to solutions visited in the most recent iterations. An important feature of tabu search is the ability to record, as the search proceeds, various information (solution values or attributes, for example) concerning the solutions encountered. This information may then be used to identify good solutions and solution regions and, more generally, to guide the exploration of the solution space. The references indicated at the beginning of this section provide numerous details on the various features that may be used to build tabu search methods.

Our tabu search method follows a rather straightforward pattern: following an initialization phase, the search proceeds with a sequence of local searches and diversification phases until a predetermined stopping criterion is met. Each local search is made up of several alternating series of pivot moves and path generation phases. Computations stop with the best solution encountered during the search (Z_{best}) once a predefined number of diversification phases (max_div) have been performed. Schematically, the procedure may be written as

1. Obtain initial solution;
   $\mathcal{C}$: Set of candidates – current paths;
   $Z_{best} = \infty; div = 0.$
2. Local Search.

3. If $div < max_div$ then

   - Perform diversification moves;
   - $div = div + 1$;
   - Go to Step 2.

4. Else STOP.

In the following, we describe the main algorithmic choices and detail the steps of the procedure.

3.1 Neighbourhoods and Moves

Formally, the solution space we want to explore is the set $\tilde{\mathcal{H}} = \bigcup_{\tilde{y} \in Y} \tilde{\mathcal{H}}(\tilde{y})$ of the extreme points of $\mathcal{H}$, defined over the set of all possible *network configurations* (design variable vectors) Y by equations (2) to (4):

$$\begin{aligned} \sum_{l \in \mathcal{L}^p} h_l^p &= w^p \quad \forall p \in \mathcal{P}, \\ \sum_{p \in \mathcal{P}} \sum_{l \in \mathcal{L}^p} h_l^p \delta_{al}^p &\leq u_a y_a \quad \forall a \in \mathcal{A}, \\ h_l^p &\geq 0 \quad \forall p \in \mathcal{P}, l \in \mathcal{L}^p. \end{aligned}$$

Two neighbourhoods are defined: a continuous neighbourhood for the local search phase, and a discrete neighbourhood used to diversify the search.

The *continuous* (or *local*) neighbourhood, $N(\tilde{h})$, of any element $\tilde{h} \in \tilde{\mathcal{H}}$, is defined as the subset of all extreme points of $\tilde{\mathcal{H}}$ adjacent to $\tilde{h}$, that is, all extreme points which can be reached from $\tilde{h}$ by one simplex pivot. Consequently, in linear programming terms, a *local move* corresponds to a transition from one basis of the system (equations (2) to (4)) to an adjacent one: a basic path thus becomes unbasic, while a previously unbasic path takes its place in the basis.

Generally, however, not all the path variables are available at each iteration. Consider the sets $\mathcal{L}_t^p = \{\text{available paths for commodity } p \text{ at iteration } t\}$, for all $p \in \mathcal{P}$, and $\mathcal{H}_t$ the corresponding polyhedron defined by equations (7) to (10)

$$\sum_{l \in \mathcal{L}_t^p} h_l^p = w^p \quad \forall p \in \mathcal{P}, \tag{7}$$

$$\sum_{p \in \mathcal{P}} \sum_{l \in \mathcal{L}_t^p} h_l^p \delta_a^p + s_a = u_a \quad \forall a \in \mathcal{A}, \tag{8}$$

$$h_l^p \geq 0 \quad \forall p \in \mathcal{P}, l \in \mathcal{L}_t^p, \tag{9}$$

$$s_a \geq 0 \quad \forall a \in \mathcal{A}, \tag{10}$$

where s_a are slack variables associated with links a. Then, $\widetilde{h}_t$, the current solution at iteration t, is an extreme point of $\mathcal{H}_t$, and there exists a subset of neighbours, $N_t(\widetilde{h}_t) \subseteq \mathcal{N}(\widetilde{h}_t)$, which are all exclusively made up of paths in $\cup_{p \in \mathcal{P}} \mathcal{L}_t^p$, $\forall p \in \mathcal{P}$, and thus are adjacent extreme points of $\mathcal{H}_t$.

To explore this neighbourhood, we partition the $\widetilde{h}_t$ solution into its basic $\widetilde{h}_B \subset \cup_{p \in \mathcal{P}} \mathcal{L}_t^p$ and non basic $\widetilde{h}_N \subset \cup_{p \in \mathcal{P}} \mathcal{L}_t^p$ components. A *candidate pivot-move* is then simply a pair of variables (h_{in}, h_{out}), $h_{in} \in \widetilde{h}_N$ and $h_{out} \in \widetilde{h}_B$, on which a simplex pivot may be performed. The *candidate list* C is the set of all such pairs of variables. Column generation may be used to enrich the set of candidates as detailed later on in the section.

The *discrete* neighbourhood is defined relative to the design variables and is used to drastically modify the network configuration and diversify the search. Formally, a discrete neighbour $\hat{h} \in D(\widetilde{h})$ of $\widetilde{h} \in \mathcal{H}$ at iteration t is the set of optimal paths in $\cup_{p \in \mathcal{P}} \mathcal{L}_t^p$ that corresponds to the network configuration $\hat{y}$ obtained by "closing" a given number of network arcs in $\widetilde{y}$. The corresponding *diversification move* is thus a complex sequence of operations which usually includes several (primal and eventually dual) pivots. Several discrete neighbourhoods may be defined (and several may be used in the same tabu search procedure) according to the number of arcs to be closed. The criteria used to select these arcs and to determine the moment when diversification moves are to be performed are fundamental design parameters of the tabu search procedure.

3.2 Local Search

The local search phase aims to explore the space of the path flow variables according to the well known principles of the revised simplex method with column generation. However, because we are not in a true linear programming context, two auxiliary MCNF formulations are defined over the constraint set (7) - (10):

1. The *Linear component* formulation, with objective function

$$\text{Minimize } z(h) = \sum_{p \in \mathcal{P}} \sum_{l \in \mathcal{L}_t^p} k_l^p h_l^p$$

which is the continuous component of the "true" objective function of the design model. Let α be the vector of dual variables associated with equations (7) and (8) of the corresponding minimum cost capacitated multicommodity network flow formulation.

2. The *Linearized* formulation, with objective function

$$\text{Minimize } z(\tilde{h}) = \sum_{p \in \mathcal{P}} \sum_{l \in \mathcal{L}_l^p} \tilde{k}_l^p h_l^p$$

where $\tilde{k}_l^p = \sum_{a \in A} \delta_{al}^p (c_a^p + f_a/u_a)$. Let $\tilde{\alpha}$ be the vector of dual variables associated with equations (7) and (8) of this formulation.

The local search phase of the tabu search procedure is then performed as follows:

Step 0. *Initialize* short term tabu lists;
C: Set of candidates – current paths;
Z_{best}: Value of the best overall solution obtained to date;
$Z_{local} = \infty$; $Z_{prev} = \infty$; $g = 0$.

Neighbour evaluation and move selection.

Step 1. Repeat steps 2 to 7 until *max_move* consecutive pivots (moves) have been executed with no improvement in the value of Z_{local}.

Step 2. Lets B identify the basis corresponding to the current solution;
Compute the values of the dual variables: $\alpha B = k_l^p$;
Pivot in any slack variable with negative marginal cost.

Step 3. Compute the reduced costs $\bar{k}_l^p$ for all non basic paths.

Step 4. For each non basic path variable i (potential h_{in} variable):

- Determine the corresponding exiting (basic) path j (potential h_{out}) variable and associated value β of the entering variable;
- Determine the corresponding modification (if any) to the vector of design variables y and compute the associated variation in the total fixed cost of the network $\Delta_{i,j}(Z(y))$;
- Compute the value of the potential move as the variation in the evaluation of $z(h, y)$, the objective function of the original network design problem: $\Delta_{i,j}(Z) = \bar{k}_i \beta + \Delta_{i,j}(Z(y))$.

Step 5. Identify $(h_{in}, h_{out}) \in argmin\{\Delta_{i,j}(Z)\}$.

Pivot-move

Step 6. If the selected move is not tabu, implement it.
If it is tabu, implement only if it improves Z_{best};
Otherwise, select next best candidate move.

Step 7. Establish and update tabu tenures;
Update long term memories;
If improved, update Z_{local} and, eventually, Z_{best}.

Column generation

Step 8. If $Z_{local} < Z_{prev}$ then $Z_{prev} = Z_{local}$ and $g = 0$;
Else $g = g + 1$ and if $g > max_col_gen$ EXIT.

Step 9. Compute the values of the dual variables for the linearized formulation:
$\tilde{\alpha}B = \tilde{k}_l^p$;
Generate new paths and add them to $\mathcal{C}$;
Go to Step 1.

To compute the value of a potential move, the procedure adds up the variations in the value of the objective function of the original network design problem (equation (1)) induced, on the one hand, by the flow variation due to the simplex pivot and, on the other hand, by the corresponding modifications to the design vector due to changes in the utilization of the network. Then, any move with $\Delta_{i,j}(Z) < 0$ is strictly improving with respect to the current solution. When no strictly improving move is available, the procedure selects the move with the largest "improvement", even if this means degrading the solution. Using the actual objective function to evaluate moves does imply an extra cost in evaluation time, but it offers both a precise ordering of potential moves and a direct aspiration criterion.

Moves are revised simplex pivots implemented according to the primal partitioning logic of [13]. Note that, when possible, slack variables are pivoted in first, before path variables are considered. Intuitively, a slack pivot is an adjustment step: flow is readjusted among the paths in the current basis such as to generate some residual capacity on a saturated arc. More formally, a slack variable with negative reduced cost corresponds to a capacity constraint with a positive dual variable, which might create a negative cycle.

Several approaches are possible to prevent the tabu search procedure from cycling. Some, such as forbidding the reversal of the last n_p pivots, might become computationally expensive; some others, such as forbidding the last n_v paths to have entered the basis to leave it, might drive the procedure into a gridlock. Also, it appears unwise to put restrictions on slack variables, since this can have similar consequences. The approach

we have chosen works on the exiting variables only, by forbidding a variable that recently got out of the basis to enter back into the basis. The *tabu tenure* of each exiting variable is randomly chosen in a [*low_tab_mv, high_tab_mv*] interval according to a discrete uniform distribution.

In a classical column generation scheme, the algorithm proceeds to price out existing non basic variables and then pivots until a local optimum is attained with respect to these variables, i.e. until all non basic variables display positive reduced costs. Then, a new, "improving" variable is generated, and the algorithm stops when this variable exhibits a positive reduced cost. In our approach, we also call upon column generation only when a local optimum has been reached relative to the paths already generated (i.e., presently in $\mathcal{C}$). However, in the present setting, one cannot use marginal costs to determine that no further improvement is possible by using the existing paths. Therefore, local optimality is defined as no improvement in the value of Z_{local} for *max_move* consecutive moves.

A shortest path algorithm over the links of the network is used to generate new paths. To capture the interplay between the fixed costs and capacities of the arcs, the reduced arc costs used by the column generation subproblem correspond to the *linearized* multicommodity capacitated network flow problem with the surrogate arc costs: $\bar{c}_a^p = c_a^p + f_a/u_a,\ \forall a \in \mathcal{A}, p \in \mathcal{P}$. In this version, all arcs are available for the generation of new paths, the dual variable associated with the capacity constraints keeping the procedure away from saturated arcs.

Several strategies are possible relative to "how many" and to "which" path variables to generate each time the column generation phase is called upon. One could, for example, generate one or several paths (in the latter case, one could generate a fixed or a variable number of paths), and one could do it for all or only for specific origins, destinations, or origin–destination pairs. The choice of a strategy impacts on the efficiency of the search, as illustrated in the next section. We chose to generate a fixed number (identified as *k_gen*) of paths for all origin–destination pairs, and to attempt to generate new paths for all O–D pairs each time the procedure is executed.

When several paths have to be generated for an O–D pair, we desire to obtain paths which are dissimilar as much as possible, without having to implement a time-consuming procedure. Our approach iteratively starts from the successors of the origin (other than the one on the shortest reduced cost path) and computes the shortest marginal cost path from each of these nodes to the destination. If all such successors are used and still the required number of paths has not been generated, one proceeds to the successor node on the initial shortest path and applies the same procedure. In all cases, the path, or paths, with the lowest reduced cost(s) is (are) then added to the set of paths $\mathcal{C}$. Note that, since this is not a classical simplex case where only variables with negative reduced cost are generated, we are not ensured that a newly generated path is not already present in $\mathcal{C}$. We implement a hashing mechanism to speed up this verification procedure.

A sequence of local neighbourhood explorations (by pivoting) followed by a column generation phase is called a *column generation cycle*. Local search is terminated when no improvement to the current local best solution is obtained after a number of consecutive column generation cycles. The method then either stops or proceeds to diversify the search (see Section 3.4). The parameter *max_col_gen* controls the termination of the local search procedure.

3.3 The Initialization Phase

Initially, a shortest path is generated for each demand using the surrogate arc costs. Demand is then sequentially loaded on these paths, the overflow being assigned to artificial paths with arbitrarily high costs. Starting from this principle, two basic initialization procedures may be devised. The first starts with all arcs open and performs simplex pivots without the tabu mechanisms until a first feasible solution is obtained to the PCMND. This method is thus equivalent to a classical Simplex Phase I. The second mechanism adopts the same Phase I philosophy, but makes use of the local search routine when looking for the first feasible solution, thus initiating the tabu logic from the very start.

Alternate strategies are possible, however, based on the hypothesis that not all arcs are required to move the traffic in the optimal solution. One may thus attempt to "guess" which arcs will not belong to the optimal configuration, and "close" (eliminate) them before starting to search for an initial solution. Several relatively simple heuristic measures based on arc characteristics may be used to determine the arcs to close. One may, for example, compute the ratio between the fixed cost and the capacity of the arcs, and close those with the highest ratio. Then, once a number of suitable arcs are closed, one of two initialization strategies described above may be applied to the resulting network. It is hoped that this will yield good initial solutions, i.e. solutions which will help find better final solutions, or accelerate the search, or both. Our initial experiments have produced inconclusive results, however. Consequently, it was decided to proceed with the direct initialization strategies described above.

3.4 Diversification

Once the completion of a local search sequence is signaled by *max_col_gen* consecutive column generation phases without improvement, the algorithm proceeds to a diversification phase. This step aims to take the search out of an apparent local optimum and toward some promising region.

The diversification strategy implemented in the current version of the algorithm is based on the observation that when using column generation to address problems such as

the multicommodity network flow and design problems, a number of "good" arcs (with respect to capacity or to fixed-cost-to-capacity ratio) appear again and again in the paths used to satisfy demand. Hence, we have implemented two simple long term *frequency* memory structures. The first records for how many iterations an arc has been in the basis, that is for how long it belonged to at least one basic path. The second frequency vector records how often an arc was part of an entering path variable. In both cases, one attempts to capture a persistence type of attribute of the design arcs, in order to better move away from current solutions. In this respect, one might expect the first approach to provide a more effective measure since it also captures a consistency attribute (for how long an arc was present "in" the basis) and somewhat filters out the arcs that are prone to get in and out of the basis in rapid succession. The experimental results have confirmed this intuition.

Arcs which score high in any one of these two memories will tend to belong to the group of arcs often used in the solutions already explored. Therefore, a solution which does not use some of them will be in a different region than the ones already visited. To diversify, one hence selects a small number of often used arcs and closes them. This is done via a procedure (akin to dual simplex pivots) which removes from the basis any path which contains at least one of the closed arcs. (This may result in an infeasible solution with flows on artificial paths; we will try to regain feasibility during the following pivoting operations.) During the tabu tenure of these arcs, all paths that contain them are not allowed to enter the basis unless, of course, the aspiration criterion overrides the tabu status. Furthermore, the closed arcs are not available during the column generation phases. The length of the tabu tenure of these arcs and paths is defined in terms of a number of column generation cycles, the *tabu_cycle*, following which the normal termination criterion of the local search procedure is activated.

4 Calibration of the Tabu Search Metaheuristic

It is well known that the performance of metaheuristics is highly dependent upon the appropriate calibration of the search parameters. A calibration phase is thus an integral part of the development of any metaheuristic procedure and ours is no exception.

The calibration process aims to determine values, or value ranges, for the search parameters such that the metaheuristic performs well over a broad range of problem types. A common pitfall of metaheuristic calibration phases is to tune too finely the procedure to a particular set of problems (or even to report performance results on the same set of problems used for calibration). To avoid this problem, we look for a set of parameters which is robust for a small set of representative problems, and then use the resulting parameter settings to experiment with two different sets of problem instances (described in Section 5).

Ten problems were selected for calibration purposes over the entire range of network sizes: from 100 to 700 design arcs and from 10 to 400 commodities. All problems display tight capacities and dominant fixed costs, initial experiments having shown that these characteristics make problems more difficult to solve.

In the following, we present the main alternatives that were analyzed and the parameter settings that were ultimately selected. The complete calibration analysis is reported in [9].

4.1 Local Search Calibration

We first focussed on the local search phase of the method and attempted to identify relationships between the attributes of a problem – its dimensions, importance of fixed costs compared to variable costs, and tightness of the capacity constraints – and the "best" values for the tabu tenure parameters. An initial set of results over a large set of problem instances did not reveal any such consistent relation. Therefore, for the rest of the calibration phase, we rather looked for a robust set of parameters, which could offer a consistent performance level over the problems tested. These initial experiments allowed us, however, to fix the value of the *max_move* parameter to twice the value of the maximum tabu tenure *high_tab_mv*.

We tested the following combinations of parameter values (see Section 3 for their definitions), without activating the diversification feature, and fixing *max_col_gen* at 10 (which allows for a long local search phase):

- Tabu tenure interval for exiting path variables [*low_tab_mv, high_tab_mv*]: [2,5], [5,10], and [10,20];
- The type of initial solution: simplex Phase I without tabu mechanisms (*YES*) or no independent Phase I and the tabu mechanisms of the local search start from the possibly infeasible initial flow allocation (*NO*);
- The number of paths generated for each commodity (origin–destination pair) during the column generation cycle: *k_gen* = 1, 2, 3, and 5.

The success of each parameter combination was measured by the number of first, second, and third best solutions it found over the problem set. Aggregating in turn for each individual parameter yielded the measures displayed in Table 1, and allowed us to gain some insight into the behaviour induced by each parameter:

- Activation of the tabu logic from the very beginning of the initialization phase appears best.

- A tabu tenure randomly chosen from a [10,20] interval performs marginally better than when the [5,10] is used. The [2,5] interval was dropped from all following experiments since is clearly dominated by the other two.
- Generating 3 or 5 paths appears to be preferable to the other two options. However, since the computation time of this phase increases with the number of generated paths, we prefer to generate only three paths for each commodity.

Parameter	**1st place**	**2nd place**	**3rd place**
Initialization procedure			
YES	5	4	3
NO	7	11	7
low_tab_mv, high_tab_mv, max_move			
2,5,10	X	X	X
5,10,20	6	6	4
10,20,40	9	6	6
k_gen			
1	4	3	5
2	3	2	1
3	4	5	2
5	4	5	2

Table 1: Individual Parameter Performance

To make the final decision, we weighted each first, second, and third places with three, two, and one points, respectively. The resulting relative performance of the various combinations of parameters is displayed in Table 2. One combination stands out and was used in all the tests that followed: *10,20,40,No,k_gen=3*, that is, no separate initialization phase, tabu tenure selected from a [10,20] iterations interval, 40 consecutive unimproving moves to decide that a current local best solution is a local optimum with respect to the existing paths, and 3 paths generated for each commodity during a column generation phase.

Setting	*k_gen=1*	*k_gen=2*	*k_gen=3*	*k_gen=5*
10,20,40,Yes	3	0	1	12
10,20,40,No	6	8	12	9
5,10,20,Yes	6	3	3	1
5,10,20,No	8	3	8	2

Table 2: Parameter Setting Performances

A further examination of these results reveal that, here as in other tabu search applications, one may find several sets of parameters which yield an efficient procedure. In

the present case, the combination *10,20,40,Yes,k_gen=5* scored as high as the one we selected (but required the generation of five paths for each product), while several others are not far behind. We take these results as an indication that the tabu search procedure is rather robust with respect to the values chosen for the local search parameters. Moreover, these combinations of parameter values may prove very valuable for efficient parallel variants of the algorithm in which several cooperating search threads use different parameter settings (see [8] for a description and analysis of such strategies).

To further document this conclusion, we examined the gaps between the best solution among all the tested parameter settings and the one yielded by the selected set of parameter values. As expected, one generally does not perform as well using the later as when the procedure is fined tuned for each specific problem. But, of course, one does not calibrate a model for every problem instance. Thus, in the present case, the selected set of parameters performs very satisfactorily over the range of test problems: the average gap between the solutions obtained by using the selected set of parameters and the best solutions is 1.2%, while the maximum observed difference is of the order of 4%. A second general remark relative to the robustness of the tabu search procedure with respect to the parameter settings, concerns its behavior when a "bad" choice of parameter values is used. By observing the differences between the best and the worst solutions, we noticed that while the maximum is of the order of 11%, most differences are significantly smaller, with an average of 4%. Therefore, even when the parameters defining the local search are relatively badly chosen, one could still expect to observe a rather good behavior of the procedure. This is important, since it is not possible to experiment with all types and sizes of networks.

4.2 Diversification

The diversification phase requires that two more parameters be considered and jointly fixed: *max_col_gen*, which controls when the local search terminates and a diversification phase is initiated, and *tabu_cycle*, the tabu tenure of the arcs closed by the diversification move. The idea here is twofold. On the one hand, one would like to avoid unnecessary column generations. To save time, of course, but also because experimentations with local search alone has shown that it is not necessary to wait for a long sequence of unimproving column generation cycles before deciding on a diversification move. On the other hand, one also desires that sufficient time is allowed to the exploration of the "new" region. It is important, in particular, not to terminate the local search phase that follows a diversification move before the arcs which have been closed are no longer tabu and a column generation phase (making use on the current dual variables which correspond to a basis where "new" paths have been introduced) can be performed with all the arcs. After a number of tests, we decided to use $max_col_gen = 3$ and $tabu_cycle = 2$.

The two diversification mechanisms are based on the idea of forcing the search in a

direction where some of the arcs currently heavily used are not part of the solution by arbitrarily "closing" them. It is what may be called a "drastic" diversification approach since, in a context where all moves are made in the space of the path variables assuming that all arcs are available, such a move may significantly alter the topology of the network. Indeed, we tried initially to close an important number of arcs, of the order of 10% and higher. We realized, however, that it was too much for most problems. Often, one could not achieve a feasible solution following a diversification move. In many other cases, the paths generated following the diversification phase were too long to compete once all arcs were made available again. Therefore, we experimented with lower values, closing 1% and 5% of the design arcs. The results indicated that, as expected, the first diversification strategy generally achieves better results than the second one (but not by a very large margin), while closing 1% of arcs clearly outperforms the alternate choice.

We also performed a simple evaluation of the worth of diversification. We compared the solution quality obtained by running the complete tabu search method until 2 diversification phases were performed to that of the local search phase allowed to run for many iterations (we actually stopped it once 10 consecutive unimproving column generation cycles were observed). We concluded that the present diversification mechanism is beneficial without yielding very big improvements. Actually, for the 42 problems of set C (Section 5), the same solutions have been found for 15 problems, while for 18 the solutions improved 1.16% in average. For only 9 problems a worse solution has been found, for an average objective value degradation of 0.08%. Consequently, our final procedure includes a diversification phase.

5 Experimentation and Analyses

The main objectives of the experimentation phase are (i) to characterize the behavior of the search algorithm where pivoting and column generation are driven according to tabu search principles; (ii) to gain insight into the performance of the method relative to three main problem instance characteristics: dimensions, relative importance of fixed costs compared to variable costs, and the degree of capacity tightness, and (iii) to establish how well the proposed algorithm solves the type of problems of interest here.

The results of the tabu search metaheuristic are compared to two other heuristics as well as to the standard branch-and-bound algorithm offered by a widely known commercial software. The tabu search procedure of Section 3, with the parameter settings of Section 4, is used for these comparisons reported in Section 5.1. Four variants of this basic algorithmic framework are analyzed in Section 5.2.

Two series of problems have been generated. These are general transshipment networks, with no parallel arcs and one commodity per origin–destination pair. On each

arc, the same unit cost is used for all commodities. Problems differ in the number of nodes, arcs, all of which are design arcs, and commodities. Several instances have been generated for each problem dimension by varying the relative importance of fixed versus variable costs and the capacity of the network compared to the total demand. The problem generators (see [15, 16] for a complete description), as well as the problem instances can be obtained from the authors.

The tabu search metaheuristic is programmed in *FORTRAN77*. Unless stated otherwise, the experiments reported in this section have been performed on SUN UltraSparc 1/140 workstations with 64MB of RAM memory.

5.1 Performance Analysis

To analyze the behavior and performance of the tabu search heuristic, we compare its output to the optimal solution obtained by using the branch-and-bound algorithm of CPLEX version 4.0, with primal simplex-based bounding [6]. We also compare the tabu search to two other heuristics. The first procedure accepts only improving (pivot) moves in Step 5. This corresponds to a classical greedy descent procedure. Contrasted to the results of the tabu search method, it acts as a measure of the impact on the solution quality of the imposition of a metaheuristic - tabu search - logic over the greedy search.

The second heuristic is based on the dual information determined when computing lower bounds on the optimal solution of the problem by Lagrangian Relaxation methods. Gendron and Crainic [15, 16] have formulated and compared several lower bounds for the fixed cost, multicommodity capacitated network design formulation and have shown the theoretical equivalence of the bounds yielded by the strong linear relaxation, the Lagrangian relaxation of the linking constraints (3), the Lagrangian relaxation of the flow constraints (2), and the corresponding Lagrangian Decomposition. When properly implemented [17, 7], these relaxations yield very good approximations of the theoretical lower bound and are generally tight (the average gap to optimality is of the order of 8% to 9% for the two sets of problems used for this study). The upper bound heuristic, that we use in the present study, makes use of the information yielded by the lower bounding procedure and combines projection and resource decomposition methods.

Tables 3, 4, and 5 display computational results for the 43 problems in the first set: identified with the letter **C**, it includes the largest problems solved. Problems are identified with a quintuplet which indicates (i) the number of nodes; (ii) the number of arcs; (iii) the number of commodities; (iv) if fixed costs are relatively high (**F**) or low (**V**) compared to variable costs; and (v) if the problem is tightly (**T**) or somewhat loosely (**L**) capacitated.

For each problem instance, the **GREEDY** and **UBR** columns display the value of

PROB	GREEDY	UBR	OPT	TABU	GAP
20,230,40,V,L	431179	431701	423848 (9.46)	425046 (71.29)	0.28%
20,230,40,F,T	4.3e+10	404638	371475 (82.27)	371816 (90.28)	0.09%
20,230,40,F,T	3.6e+09	679539	643036 (1639.34)	644172 (121.79)	0.17%
20,230,200,V,L	146464	164770	94752 (t)	122592 (504.50)	29.38%
20,230,200,F,L	232817	299590	139888 (t)	188590 (491.63)	34.82%
20,230,200,V,T	136978	204486	98051 (t)	118057 (548.36)	20.40%
20,230,200,F,T	6.8e+06	277365	137796 (t)	182829 (889.69)	32.68%
20,300,40,V,L	431839	447235	429398 (1.25)	429912 (71.05)	0.12%
20,300,40,F,L	624978	734217.1	586077 (188.53)	589190 (113.44)	0.53%
20,300,40,V,T	3.2e+10	481738	464509 (176.91)	464509 (145.33)	0%
20,300,40,F,T	1.2e+11	650874	604198 (71.45)	606364 (123.42)	0.35%
20,300,200,V,L	104195	168510	75460 (t)	88398 (982.21)	17.15%
20,300,200,F,L	205254	300507	116810 (t)	151317 (1316.75)	29.54%
20,300,200,V,T	101772	162044	75090 (t)	82724 (938.29)	10.17%
20,300,200,F,T	2.5e+08	302376	112650 (t)	135593 (1065.88)	0.40%

Table 3: Computational Results I - C Problems

the best feasible solution obtained by the greedy descent and the resource-decomposition heuristic, respectively. The **OPT** column corresponds to the branch-and-bound algorithm of CPLEX version 4.0, with primal simplex-based bounding. To obtain the values displayed in the **TABU** column, the setting of parameters indicated at the conclusion of Section 4 was used for all problems. In all tables, a **X** indicates that the procedure has failed to find a feasible solution, while a **t** indicates that the procedure stopped due to a time limit condition. When the branch-and-bound has identified a feasible solution, the **GAP** column displays the *optimality gap* of the tabu procedure relative to it.

Initial experimentation had indicated that the branch-and-bound has difficulties with the larger problems: very large enumeration trees and significant computation times. Consequently, we ran both the branch-and-bound and the tabu metaheuristic on a SUN UltraSparc-II workstation (2 CPUs, but only one allocated to our experiments), with a 296MHz clock, 2 Mb of cache memory, and 2 Gb of RAM memory. A limit of 6 hours (21000+ seconds) computation time was imposed on the branch-and-bound. Computation times in Tables 3 to 5 are indicated in seconds of CPU time.

The results illustrate the complexity of capacitated multicommodity network design problems and the limitations of standard methods and state-of-the-art software packages to address them. Thus, after more than six hours of computation, a standard branch-and-bound cannot identify the optimal solution for all (but one) problem instances with more than 200 commodities. Moreover, for most cases with 400 or more commodities, it cannot even identify a feasible solution.

The greedy heuristic fails miserably. And the feasible solutions obtained by the resource decomposition heuristic are far from being of acceptable quality. For a number of problem instances (6 out of 186 [9]),the procedure fails to identify a feasible solutions. Moreover, compared to the tabu search method, solutions are always worst, from 1% to almost 250% higher, for an average gap of 73.50%. In comparison, the tabu search metaheuristic finds solutions to all problems and generally cuts the optimality gap by half. Moreover, the quality of the solutions identified by the resource decomposition heuristic is very sensitive to the size of the problem: It particularly degrades with the number of commodities. Thus, the largest gaps between the resource decomposition and tabu search heuristics are found for the 400 commodities problems: From 80.8% to 248.9%, for a 196.58% average.

The tabu search metaheuristic appears significantly more robust with respect to the problem dimensions and number of commodities than both the resource decomposition heuristic and the branch-and-bound method. Yet, it is not totally independent of these influences, either. A second set of problems has been generated, therefore, in order to further study these relationships. The problem instances in this set, identified with the letter **R**, have been designed to facilitate the evaluation of the impact on performance of one problem characteristic only. Thus, the number of design arcs and commodities are

PROB	GREEDY	UBR	OPT	TABU	GAP
30,520,100,V,L	4.6e+08	67699	54104 (t)	56426 (995.64)	4.29%
30,520,100,F,L	1.0e+09	221188	96450 (t)	104117 (939.24)	7.94%
30,520,100,V,T	2.6+09	75757	52258 (t)	53288 (1218.52)	1.97%
30,520,100,F,T	115244	185746	99870 (t)	107894 (670.29)	8.03%
30,520,400,V,L	1.9e+10	226070	113021 (t)	125831 (5789.27)	11.33%
30,520,400,F,L	3.0+08	389636	X (t)	177409 (6406.62)	-
30,520,400,V,T	3.1e+11	X	X (t)	125518 (6522.23)	-
30,520,400,F,T	2.0e+10	336248	X (t)	174526 (8415.24)	-
30,700,100,V,L	51962	59411	47603 (1775.69)	48984 (1265.11)	2.90%
30,700,100,F,L	73892	114603	60559 (t)	65356 (1479.59)	7.92%
30,700,100,V,T	2.8e+09	60918	46564 (t)	47083 (2426.02)	1.12%
30,700,100,F,T	64003	97023	55709 (t)	58804 (1735.72)	5.56%
30,700,400,V,L	1.3e+10	215014	X (t)	110000 (12636.20)	-
30,700,400,F,L	1.8e+10	387566	X (t)	165484 (11367.70)	-
30,700,400,V,T	2.0e+10	202740	X (t)	103768 (15879.50)	-
30,700,400,F,T	2.12e+10	375586	X (t)	150919 (11660.40)	-

Table 4: Computational Results II - C Problems

PROB	GREEDY	UBR	OPT	TABU	GAP
25,100,10,V,L	14763	14828	14712 (0.12)	14712 (5.60)	0%
25,100,10,F,L	17073	20182	14941 (91.47)	15889 (8.37)	6.35%
25,100,10,F,T	1.2e+07	57314	49899 (441.86)	51654 (17.10)	3.52%
25,100,30,V,T	6.1e+09	382923	365272 (20.35)	365272 (16.57)	0%
25,100,30,F,L	6.9e+06	48782	37055 (15274.21)	38804 (33.01)	4.72%
25,100,30,F,T	3.0e+08	99896	85530 (2283.71)	86445 (71.84)	1.07%
100,400,10,V,L	6.4e+07	30273	28433 (775.40)	28485 (32.66)	0.18%
100,400,10,F,L	28279	45188	25166 (t)	24912 (33.00)	-1.00%
100,400,10,F,T	2.4e+07	109244	71439 (t)	71128 (81.23)	-0.44%
100,400,30,V,T	9.0e+09	407085	385102 (383.77)	385185 (277.50)	0.02%
100,400,30,F,L	63000	110401	52368 (t)	58773 (100.16)	12.23%
100,400,30,F,T	1.7e+09	226054	145083 (t)	149282 (215.71)	2.89%

Table 5: Computational Results III - C Problems

Prob. Set	$\leq 0\%$	$\leq 1\%$	$\leq 5\%$	$\leq 10\%$	$\leq 20\%$	$> 20\%$
C	12	8	8	5	4	6
R	45	19	52	24	8	5
All	57	27	60	29	12	11

Table 6: Distribution of Relative Gaps

Prob. Set	X	Opt	Improvement Gap	Strict Opt Gap	Worst
C	7	3	0.72% ($n = 2$)	8.97% ($n = 31$)	34.82%
R	15	16	8.95% ($n = 14$)	5.45% ($n = 108$)	28.85%
All	22	19	6.56% ($n = 16$)	6.24% ($n = 139$)	

Table 7: Aggregate Relative Gaps

systematically varied and, for each combination of dimensions, nine problem instances are generated to account for various levels of variable to fixed cost and total demand to total capacity ratios. A high fixed cost ratio indicates that fixed costs are relatively important compared to the total transportation cost of the problem instance. A high capacity ratio indicates that the total capacity of the network is relatively low compared to the total commodity demand. Three fixed cost - $F01 = 0.01$, $F05 = 0.05$, and $F10 = 0.10$ - and three capacity - $C1 = 1$, $C2 = 2$, and $C8 = 8$ - ratios are used. Thus, the "easiest" problems generally have $F01$ and $C1$ ratios, while the "most difficult" display $F10$ and $C8$ characteristics. The full result tables are presented in [9]. In order not to overload the paper, only aggregated measures are included herein.

Comparing the results obtained by the tabu search method to the solutions of the CPLEX branch-and-bound, one gets the results displayed in Tables 6 and 7. Table 6 displays the number of problem instances in each of the relative gap intervals. The figures in the first column ($\leq 0\%$) correspond to all instances for which either the optimum solution has been identified or the tabu procedure improved over the branch-and-bound. The next column corresponds to the $(0\%, 1\%]$ interval, and so on. These results are further qualified in Table 7. Thus, for 22 problem instances, the branch-and-bound did not find a feasible solution (column **X**). Over the 163 other problems , the tabu search heuristic identified 19 optimum solutions (column **Opt**) and improved the branch-and-bound feasible solution for 16 other problems (the **Improvement Gap** column). For the 139 other problem instances, the tabu search method achieved an average gap relative to the branch-and-bound solution of 6.24% (the **Strict Opt Gap** column). Even though the tabu procedure had difficulties with a few problems (in 5.6% cases, the optimality gap is greater than 20%), one obtains a global average optimality gap of 5.49% over the 158 problems for which the tabu search solution is greater or equal to the branch-and-bound one. This constitutes a very good performance, especially given the dimensions of most problems in set **C** and the inherent difficulty built into most **R** problems.

	C1	C2	C8
F01	2.12%	1.57%	1.65%
F02	4.70%	4.90%	3.44%
F10	10.90%	7.12%	6.01%

Table 8: Gap Distribution According to Fixed Cost and Capacity Ratios

Table 8 displays the optimality gap distribution for problems in set **R** according to the relative importance of the fixed costs and capacities. These figures appear to indicate that the tabu search metaheuristic is more sensitive to the importance of the fixed costs than to the tightness of the capacity restrictions. These results are consistent with the generally accepted idea that fixed costs play an important role in widening the gap for capacitated problems. It also appears that, for a given level of fixed cost amplitude, the tabu search method performs better when capacity restrictions are more binding (e.g., a C8 capacity ratio). A possible explanation of this behavior may be that, when the

$\|\mathcal{P}\|$	$\|\mathcal{N}\|, \|\mathcal{A}\|$	Opt. Gap	$\|\mathcal{N}\|, \|\mathcal{A}\|$	Opt. Gap	$\|\mathcal{N}\|, \|\mathcal{A}\|$	Opt. Gap
10		0.62%		1.81%		2.04%
25	10,25	0.57%	10,50	2.78%	10,75	3.13%
50		0.54%		6.62%		8.94%
40		0.97%		7.26%		6.52%
100	20,100	4.76%	20,200	5.99%	20,300	11.96%
200		5.59%		5.89%		6.19%

Table 9: Gap Distribution According to Problem Dimensions

problem is tight, the surrogate function used to compute the reduced arc costs in the path generation phase yields a better approximation of the interplay between fixed cost and capacity. Another contributing factor is the relatively low number of feasible flow patterns that correspond to very tight capacity constraints. Similar results are observed for the **C** problems.

Table 9 displays the relative optimality gaps for the **R** problems according to their dimensions: number of nodes, design arcs, and commodities. We aim to highlight and analyze the impact of the last two network characteristics. The analysis is difficult because, here as well as for problems in set **C**, the figures do not capture the instances for which an optimal or feasible branch-and-bound solution could not be found, nor those for which the tabu search identified a better solution. It appears, however, that for small problems, a significant increase in the number of commodities signals a degradation in solution quality. The impact is less noticeable for larger problems. The behavior of the procedure relative to the number of design arcs follows a similar pattern, but of reduced amplitude.

The results displayed in this section clearly show that the tabu search heuristic we propose is very effective for the class of problems studied. For all problems, the feasible solutions we obtain improve greatly on those generated by the resource decomposition heuristic. Compared to the solutions obtained by branch-and-bound, the procedure yields very good solutions in significantly less time. Moreover, the tabu search heuristic identifies good quality solutions even when a classical branch-and-bound fails due to time or memory limitations.

An interesting observation is that the tabu search procedure seems to perform better, relatively speaking, when capacities are tight. Another interesting observation is that the procedure has no more difficulty (although it takes somewhat longer) to identify good solutions for large problems with numerous commodities than it has for smaller ones. This is a significant difference with traditional heuristic approaches.

One final note before continuing on to the final phase of the experiments. The procedure presented in this paper, similarly to most adaptive search heuristics, requires a

relatively high number of iterations in order to yield good solutions. This translates into significant computation times. It is not necessary, however, to let the procedure run for as long as we did for the specific purposes of the experimental phase. Actually, good solutions are attained after a relatively small number of column generation phases. Hence, while the tabu search procedure will take longer than bounding procedures based on "exact" mathematical programming principles, the computation times could be substantially reduced for actual applications. Furthermore, for all tests, the tabu search procedure started with no initial solution. It is believed that a warm start, by using the information generated by the relaxation lower bound, for example, could contribute significantly towards reducing the computation effort. This could prove very valuable within the framework of a branch-and-bound algorithm.

5.2 Evaluation of Variants

A number of variants may be imagined around the basic search framework described in Section 3. The following were developed and tested:

DISC The selection of the diversification arcs is modified to account for the level of residual capacity available in the network once a given arc is removed. The procedure proceeds greedily as follows. Once an arc is selected, according to a criterion based on the frequency memory, one computes for each one of the concerned O-D pairs the amount of flow that has to be diverted and checks whether there is enough spare capacity in the rest of the network (by using the existing paths) to carry it. If there is not, the arc is not closed and the next one is examined. This diversification strategy does not guarantee that the solution resulting from a diversification move will always be feasible, but it aims to decrease the occurrence of infeasible solutions.

DIVA This is another diversification strategy where frequency memories are ignored and the selection of the diversification arcs is based on a random process: a given percentage (1% for the tests reported in this paper) of basic paths are selected and the arcs along these paths are closed.

SAT In a few isolated cases (four of the 195 problems tested), the paths output by the column generation procedure are not sufficiently different from the existing ones (even when the number of paths generated at each iteration is significantly increased). This behavior has been observed for very tightly capacitated problems where a limited number of arcs are part of the best paths for many commodities. To increase the difference between the new and existing paths, the shortest path routine of the column generation phase (Step 9) is therefore modified to ignore saturated arcs.

SATDISC A strategy that combines the SAT column generation approach and the DISC diversification method.

Full results for the problems in set C are presented in [9]. These results lead to the following conclusions.

- When the SAT strategy is used in the column generation phase, the type of diversification used, based on the residual capacity or on the frequency memories, does not seem to play an important role in determining the behavior of the search: SAT and SATDISC identify the same solution in about 93% of the problems (40 out of 43), while SATDISC identifies slightly better solutions in some 9% (4 problems) of problem instances. Computing times are similar.
- The basic tabu search dominates both these two strategies: it identifies better solutions for 63% of problem instances, including, in particular, all "large" (in terms of number of arcs and commodities) problems.
- Comparing the three diversification strategies - based on frequency memories (the basic strategy), network residual capacity (DISC), and random selection of arcs (DIVA) - one realizes that their behaviour is similar in most cases (70% of problem instances), the basic tabu search approach identifying equal or superior quality solutions for 18.6% of the problems. The random diversification procedure is generally more time consuming.

Two main conclusions emerge out of these experiments. First, it appears that the column generation strategy that avoids saturated arcs does not perform as well as the basic tabu search strategy. Yet, in some cases, it helps diversify the search and guide it toward good quality solutions. We propose therefore an adaptive procedure where the SAT strategy is activated (and the local search phase is restarted) when the best overall solution at the time of the first diversification phase is infeasible (this procedure has been used to obtain solutions for the four problems mentioned earlier [9]).

The second conclusion addresses the issue of the importance of the diversification phase. The study of the variants has confirmed the conclusions of the calibration phase: while the impact of diversification moves on solution quality may appear somewhat limited, they are essential to reach best solutions for most problems. Several diversification mechanisms appear to yield similar quality results. Diversification based on frequency memories appears as good as or better than the other strategy we tested.

6 Conclusions

The procedure presented in this paper represents, at least as far as we know, one of the rare successful efforts to propose an efficient approach to obtaining good feasible solutions to large, fixed charge, capacitated, multicommodity network flow problems. These are

difficult problems and realistically sized instances cannot be solved by commercial software packages. Even identifying good feasible solutions appears to be quite a challenge and the difficulty increases significantly with the number of commodities considered.

The tabu search approach we propose is different from the more classical ones where the space of the integer variables is explored (see [10] for example) and which has not, as yet, achieved significant results for the class of problems at hand. It is also one of the very few attempts to devise efficient tabu search-based procedures for mixed integer programming problem that display both a very strong combinatorial nature and a difficult linear programming component. Furthermore, the ideas of pivoting exclusively in the space of the path variables and of combining local tabu search and column generation are applied for the first time in the context of Tabu Search.

More importantly, the simplex-base tabu search metaheuristic identifies very good solutions within reasonable computing efforts. The procedure significantly outperforms relaxation-based heuristics in terms of solution quality and the relative optimality gaps are low. Moreover, the procedure appears robust with respect to the type of problem, in terms of relative importance of the fixed costs and capacities, size and, especially, the number of commodities.

Interesting research avenues are now possible. One such direction concerns the application of the approach presented here to other types of network design problems. A second attracting subject is a study of various other diversification strategies, in particular, methods where the path information is more fully used. The combination of the relaxation-based lower bound and the tabu search strategy within a branch-and-bound framework constitutes another fascinating research direction. And, of course, given the dimensions of most realistic applications, the development of parallel versions gets a high priority. We are active working on these subjects and hope to report on our results in the not too distant future.

7 Acknowledgments

We would like to thank our colleague, Dr. Patrick Soriano, for his valuable comments. We also thank Daniel Hébert who did most of the programming and Francois Guertin who bravely took his relay, as well as Nathalie Talbot, Alexandre Le Bouthillier, and Dominique Tourillon who performed the innumerable tests required by this project. We also desire to thank the associate Editor and the anonymous referees. Their comments have helped us improve the presentation of the paper.

Funding for this project has been provided by the Natural Sciences and Engineering Council of Canada, through its Research and Strategic Grant programs, by the Fonds

F.C.A.R. of the Province of Québec, and by the Québec–Ontario Cooperation Program.

References

[1] Ahuja, R.K., Magnanti, T.L., and Orlin, J.B. *Network Flows – Theory, Algorithms, and Applications.* Prentice-Hall, Englewood Cliffs, NJ., 1993.

[2] Assad, A.A. Multicommodity Network FLows – A Survey. *Networks*, 8:37–91, 1978.

[3] Balakrishnan, A. *Valid Inequalities for the Network Design Problem with an Application to LTL Consolidation Problem.* PhD thesis, Sloan School of Management, Massachusetts Institute of Technology, 1984.

[4] Balakrishnan, A., Magnanti, T.L., and Mirchandani, P. Network Design. In M. Dell'Amico, F. Maffioli, and S. Martello, editors, *Annotated Bibliographies in Combinatorial Optimization*, pages 311–334. John Wiley & Sons, New York, NY, 1997.

[5] Balakrishnan, A., Magnanti, T.L., and Wong, R.T. A Dual–Ascent Procedure for Large–Scale Uncapacitated Network Design. *Operations Research*, 37(5):716–740, 1989.

[6] CPLEX. CPLEX Documentation. Technical report, CPLEX Optimization, Inc., 1993.

[7] Crainic, T.G., Frangioni, A., and Gendron, B. Bundle-Based Relaxation Methods for Multicommodity Capacitated Network Design. Publication 98-45, Centre de recherche sur les transports, Université de Montréal, Montréal, QC, Canada, 1998.

[8] Crainic, T.G. and Gendreau, M. Cooperative Parallel Tabu Search for Capacitated Network Design. Publication CRT-98-71, Centre de recherche sur les transports, Université de Montréal, Montréal, QC, Canada, 1998.

[9] Crainic, T.G., Gendreau, M., and Farvolden, J.M. A Simplex-Based Tabu Search Method for Capacitated Network Design. Publication CRT-98-37, Centre de recherche sur les transports, Université de Montréal, Montréal, QC, Canada, 1998.

[10] Crainic, T.G., Gendreau, M., P. Soriano, and Toulouse, M. A Tabu Search Procedure for Multicommodity Location/Allocation with Balancing Requirements. *Annals of Operations Research*, 41:359–383, 1993.

[11] Crainic, T.G. and Rousseau, J.-M. Multicommodity, Multimode Freight Transportation: A General Modeling and Algorithmic Framework for the Service Network Design Problem. *Transportation Research B: Methodology*, 20B:225–242, 1986.

[12] Farvolden, J.M. and Powell, W.B. Subgradient Methods for the Service Network Design Problem. *Transportation Science*, 28(3):256–272, 1994.

[13] Farvolden, J.M., Powell, W.B., and Lustig, I.J. A Primal Partitioning Solution for the Arc–Chain Formulation of a Multicommodity Network Flow Problem. *Operations Research*, 41(4):669–694, 1992.

[14] Gendron, B. *Modèles et algorithmes pour problèmes de planification de réseaux et de localisation.* PhD thesis, Département d'informatique et recherche opérationnelle, Université de Montréal, 1985.

[15] Gendron, B. and Crainic, T.G. Relaxations for Multicommodity Network Design Problems. Publication CRT-965, Centre de recherche sur les transports, Université de Montréal, Montréal, QC, Canada, 1994.

[16] Gendron, B. and Crainic, T.G. Bounding Procedures for Multicommodity Capacitated Network Design Problems. Publication CRT-96-06, Centre de recherche sur les transports, Université de Montréal, Montréal, QC, Canada, 1996.

[17] Gendron, B., Crainic, T.G., and Frangioni, A. Multicommodity Capacitated Network Design. In B. Sansó and P. Soriano, editors, *Telecommunications Network Planning*, pages 1–19. Kluwer, Norwell, MA, 1998.

[18] F. Glover. Future Paths for Integer Programming and Links to Artificial Intelligence. *Computers & Operations Research*, 1(3):533–549, 1986.

[19] F. Glover. Tabu Search - Part I. *ORSA Journal on Computing*, 1(3):190–206, 1989.

[20] F. Glover. Tabu Search - Part II. *ORSA Journal on Computing*, 2(1):4–32, 1990.

[21] Glover, F. and Laguna, M. *Tabu Search.* Kluwer, Norwell, MA, 1997.

[22] Helgason, R.V. *A Lagrangean Relaxation Approach to the Generalized Fixed Charge Network Flow Problem.* PhD thesis, Southern Methodist University, 1980.

[23] Jarvis, J.J. and O. Mejia de Martinez. A Sensitivity Analysis of Multicommodity Network Flows. *Transportation Science*, 11(4):299–306, 1977.

[24] Kennington, J.L. A Survey of Linear Cost Multicommodity Network FLows. *Operations Research*, 26:209–236, 1978.

[25] Koskosidis, Y.A., Powell, W.B., and Solomon, M.M. An Optimization–based Heuristic for Vehicle Routing and Scheduling with Soft Time Windows Constraints. *Transportation Science*, 26:69–85, 1992.

[26] Lamar, B.W., Sheffi, Y., and Powell, W.B. A Capacity Improvement Lower Bound for Fixed Charge Network Design Problems. *Operations Research*, 38(4):704–710, 1990.

[27] Magnanti, T.L. and Wolsey, L.A. Optimal Trees. In Ball, M., Magnanti, T.L., Monma, C.L., and Nemhauser, G.L., editors, *Network Models*, volume 7 of *Handbooks in Operations Research and Management Science*, pages 503–615. North-Holland, Amsterdam, 1995.

[28] Magnanti, T.L. and Wong, R.T. Network Design and Transportation Planning: Models and Algorithms. *Transportation Science*, 18(1):1–55, 1986.

[29] Minoux, M. Network Synthesis and Optimum Network Design Problems: Models, Solution Methods and Applications. *Networks*, 19:313–360, 1986.

[30] Powell, W.B. A Local Improvement Heuristic for the Design of Less-than-Truckload Motor Carrier Networks. *Transportation Science*, 20(4):246–357, 1986.

[31] Rardin, R.L. Tight Relaxations of Fixed Charge Network Flow Problems. Report J-82-3, School of Industrial Engineering, Purdue University, 1982.

[32] Rardin, R.L. and Choe, U. Tighter Relaxations of Fixed Charge Network Flow Problems. Report J-79-18, School of Industrial Engineering, Purdue University, 1982.